

743

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re the application of:

Hans Albrecht SCHMID

Examiner:

Serial No.: 09/905,184

Group Art Unit: 2152

Filed : July 16, 2001

For : PROCESS FOR OPERATING A  
DISTRIBUTED COMPUTER NETWORK  
COMPRISING SEVERAL DISTRIBUTED  
COMPUTERS

Received

OCT 19 2001

Technology Center 2100

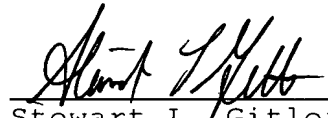
SUBMISSION OF PRIORITY DOCUMENT

Commissioner of Patents and Trademarks  
Washington, DC 20231

Sir:

Applicant submits a certified copy of German application 101 24 930.6, filed May 21, 2001. Priority of this German application is claimed under 37 U.S.C. §119. With submission of this certified copy, the claim of priority is perfected.

Respectfully submitted,



Stewart L. Gitler  
Reg. No. 31,256

October 19, 2001  
(703)415-0100

HOFFMAN, WASSON & GITLER, PC  
2361 Jefferson Davis Highway  
Suite 522  
Arlington, VA 22202  
(703) 415-0100

Attorney's Docket: A-7498.SPD/cat



**Prioritätsbescheinigung über die Einreichung  
einer Patentanmeldung**

**Aktenzeichen:** 101 24 930.6

**Anmeldetag:** 21. Mai 2001

**Anmelder/Inhaber:** Prof. Dr. Hans Albrecht S c h m i d , Allensbach/DE

**Bezeichnung:** Verfahren zum Betreiben eines verteilten Rechnernetz-  
werks umfassend mehrere verteilt angeordnete Rechner

**IPC:** G 06 F 15/16

Die angehefteten Stücke sind eine richtige und genaue Wiedergabe der ursprüng-  
lichen Unterlagen dieser Patentanmeldung.

München, den 1. Oktober 2001  
Deutsches Patent- und Markenamt  
Der Präsident  
Im Auftrag

Hiebinger

F:\IJBDHF\DHFANM\3988001

Anmelder:

Prof. Dr. Hans Albrecht Schmid  
Im Vogelgesang 2a  
78476 Allensbach

3988001

21.05.2001  
wrz / wrz

**Titel: Verfahren zum Betreiben eines verteilten  
Rechnernetzwerks umfassend mehrere verteilt  
angeordnete Rechner**

**Beschreibung**

Die vorliegende Erfindung betrifft ein Verfahren zum Betreiben eines verteilten Rechnernetzwerks umfassend mehrere verteilt angeordnete Rechner. Auf einem der Rechner ist mindestens eine auf einem Mikroprozessor des Rechners ablauffähige Komponente eines Computerprogramms angeordnet. Zum Betreiben des Rechners wird von einem kollozierten Klient oder einem entfernten Klient aus auf die Komponente zugegriffen. Der kollozierte Klient ist auf dem gleichen Rechner abgelegt und läuft innerhalb der gleichen Laufzeitumgebung (sog. execution environment) ab wie die Komponente. Der entfernte Klient ist

auf einem anderen Rechner abgelegt und/oder läuft innerhalb einer anderen Laufzeitumgebung ab als die Komponente.

Die Erfindung betrifft außerdem ein Verfahren zum Betreiben eines Rechners eines verteilten Rechnernetzwerks umfassend den Rechner und mehrere weitere verteilt angeordnete Rechner. Auf dem Rechner ist mindestens eine auf einem Mikroprozessor des Rechners ablauffähige Komponente eines Computerprogramms angeordnet. Zum Betreiben des Rechners wird von einem kollozierten Klient oder einem entfernten Klient aus auf die Komponente zugegriffen.

Die Erfindung betrifft des weiteren ein auf Mikroprozessoren von Rechnern eines verteilten Rechnernetzwerks ablauffähiges Computerprogramm. Das Computerprogramm weist mindestens eine Komponente mit mindestens einem Zugang zum Zugriff auf die Komponente von einem kollozierten Klient oder von einem entfernten Klient aus auf.

Die vorliegende Erfindung betrifft außerdem ein Speicherelement für einen Rechner eines verteilten Rechnernetzwerks. Auf dem Speicherelement ist mindestens eine Komponente eines auf Mikroprozessoren von Rechnern des Rechnernetzwerks ablauffähigen Computerprogramms abgespeichert. Das Computerprogramm weist mehrere Komponenten mit jeweils mindestens einem Zugang zum Zugriff auf die Komponente von einem kollozierten Klient oder entfernten

Klient aus auf. Als Speicherelement kann insbesondere ein Read-Only-Memory, ein Random-Access-Memory oder ein Flash-Memory zur Anwendung kommen.

Die Erfindung betrifft außerdem einen Rechner eines verteilten Rechnernetzwerks. Der Rechner umfasst ein Speicherelement, insbesondere ein Read-Only-Memory, ein Random-Access-Memory oder ein Flash-Memory, auf dem mindestens eine Komponente eines auf Mikroprozessoren von Rechnern des Rechnernetzwerks ablauffähigen Computerprogramms abgespeichert ist. Das Computerprogramm weist mehrere Komponenten mit jeweils mindestens einem Zugang zum Zugriff auf die Komponente von einem kollozierten Klient oder entfernten Klient aus auf.

Schließlich betrifft die vorliegende Erfindung ein verteiltes Rechnernetzwerk umfassend mehrere Rechner. Die Rechner weisen jeweils ein Speicherelement, insbesondere ein Read-Only-Memory, ein Random-Access-Memory oder ein Flash-Memory, auf. Auf dem Speicherelement ist mindestens eine Komponente eines auf Mikroprozessoren der Rechner des Rechnernetzwerks ablauffähigen Computerprogramms abgespeichert. Das Computerprogramm weist mehrere Komponenten mit jeweils mindestens einem Zugang zum Zugriff auf die Komponente von einem kollozierten Klient oder entfernten Klient aus auf.

Aus dem Stand der Technik sind verteilte Komponenten bspw. als sog. Enterprise Java Beans (EJB) oder Common Object Request

Broker Architecture (CORBA) verteilte Objekte oder Komponenten bekannt. Bei den bekannten Verfahren zum Abarbeiten eines aus verteilten Komponenten aufgebauten Computerprogramms auf den Rechnern eines verteilten Rechnernetzwerks umfassen die einzelnen Komponenten jeweils verschiedene Funktionalitäten der Systemumgebung (z.B. EJB oder CORBA) und anwendungsspezifische Funktionalitäten. Das verteilte Rechnernetzwerk besteht aus verschiedenen Rechnern oder Rechnerknoten. Auf einem Rechner oder Rechnerknoten ist mindestens eine Software-Laufzeitumgebung realisiert. Das Rechnernetzwerk hat bspw. eine Klient-Server-Architektur.

Zur Ausführung bestimmter anwendungsspezifischer Funktionen im Rahmen der Abarbeitung des Computerprogramms kann ein Klient auf eine bestimmte Komponente, die die gewünschten anwendungsspezifischen Funktionalitäten aufweist, zugreifen. Der Klient kann bspw. als eine weitere Komponente des gleichen Computerprogramms oder als ein beliebig anderes Computerprogramm ausgebildet sein. Wenn der Klient und die aufgerufene Komponente auf dem gleichen Rechner und innerhalb der gleichen Laufzeitumgebung realisiert sind, greift der Klient auf die Komponente im Rahmen eines sog. kollozierten (collocated) Aufrufs zu. Das bedeutet, dass der Klient und die Komponente zwar lokal zueinander angeordnet sind, tatsächlich aber der Zugriff auf die Komponente praktisch wie ein sog. entfernter Zugriff behandelt wird. Anderenfalls greift der Klient im Rahmen eines entfernten (remote) Zugriffs auf die

Komponente zu. Als Zugriff auf eine Komponente wird die Anforderung eines bestimmten Dienstes, der Aufruf einer Funktion oder Methode oder das Schicken einer Nachricht bezeichnet. Dabei können Parameter und/oder Ergebnisse übertragen werden.

Wie bereits erwähnt, wird bei dem Stand der Technik ein kollozierter Zugriff auf eine Komponente praktisch wie ein entfernter Zugriff behandelt. Ein entfernter Zugriff benötigt jedoch wesentlich mehr Abarbeitungszeit als ein lokaler Zugriff, da im Rahmen des entfernten Zugriffs u.a.

Transformationen und Rücktransformationen von Informationen zum Zwecke der Datenübertragung von dem einem Rechner, auf dem der Klient realisiert ist, zu einem weiteren Rechner des Rechnernetzwerks, auf dem die Komponente realisiert ist, durchgeführt werden müssen. Bei einem kollozierten Zugriff auf eine auf dem gleichen Rechner und in der gleichen Laufzeitumgebung angeordnete Komponente können praktisch alle Funktionen, die für einen entfernten Zugriff erforderlich sind, entfallen.

Die Behandlung von kollizierten Zugriffen wie entfernte Zugriffe beim Stand der Technik hat ihre Ursache darin, dass die bekannten Komponenten über eine entfernte Schnittstelle (sog. remote interface) mit lediglich einem entfernten Zugang (sog. remote gate) verfügen. Dadurch kann zwar eine Ortstransparenz (sog. location transparency) gewährleistet

werden, d. h. die gleiche Komponente kann ohne Umprogrammierung des Klienten sowohl kolloziert als auch entfernt aufgerufen werden. Eine Komponente mit Ortstransparenz kann realloziert und ohne weiteres von einem lokalen auf einen entfernten Rechner bzw. Rechnernetzwerknoten und umgekehrt verlagert werden. Aufgrund der entfernten Schnittstelle müssen jedoch selbst kollozierte Zugriffe auf die Komponente über den entfernten Zugang erfolgen, d. h. nahezu wie entfernte Zugriffe behandelt werden und erfordern entsprechend viel Abarbeitungszeit. Ein zeitsparender lokaler Zugriff auf eine verteilte Komponente ist beim Stand der Technik also nicht möglich.

Zwar werden bei den aus dem Stand der Technik bekannten Verfahren zum Betreiben eines Rechners oder eines Rechnernetzwerks beschränkte Optimierungen der kollozierten Zugriffe durchgeführt. Durch diese Optimierungen kann die Zugriffszeit bei kollozierten Zugriffen gegenüber entfernten Zugriffen zwar verringert werden, liegt jedoch immer noch um Größenordnungen über der für einen lokalen Zugriff.

Bei einer als Collocation Optimization bezeichneten Optimierung der kollozierten Zugriffe, wird bei einem kollozierten Zugriff auf eine Komponente der entfernte Zugang der Komponente umgangen und direkt in die Komponente gesprungen. Außerdem fällt durch den direkten Zugriff auf die Komponente die Systemfunktionalität des entfernten Zugangs weg



und muß erst wieder durch spezielle Maßnahmen aufgerufen werden. Diese Optimierung ist weiterhin nicht für Zugriffe auf Komponenten geeignet, bei denen Referenzen auf Komponenten als Parameter oder Ergebnisse übergeben werden müssen. Bei dieser Optimierung verfügen die Komponenten also nur noch über eine beschränkte Ortstransparenz (location transparency).

Der vorliegenden Erfindung liegt deshalb die Aufgabe zugrunde, das Abarbeiten eines auf Mikroprozessoren von Rechnern eines verteilten Rechnernetzwerks ablauffähigen Computerprogramms mit mehreren Komponenten zu beschleunigen.

Zur Lösung dieser Aufgabe wird ausgehend von dem Verfahren zum Betreiben eines Rechnernetzwerks und von dem Verfahren zum Betreiben eines Rechners der eingangs genannten Art vorgeschlagen, dass auf die Komponente von dem kollozierten Klient aus über einen lokalen Zugang (local gate) der Komponente zugegriffen wird, falls der Klient auf dem gleichen Rechner abgelegt ist und innerhalb der gleichen Laufzeitumgebung abläuft wie die Komponente, und anderenfalls von dem entfernten Klient aus über einen entfernten Zugang (remote gate) der Komponente auf die Komponente zugegriffen wird.

Die Komponenten weisen eine oder mehrere Schnittstellen (sog. interfaces) auf, wobei also für jede Schnittstelle zwei getrennte Zugänge (sog. gates) vorgesehen sind, ein lokaler

Zugang (sog. local gate) und ein entfernter Zugang (sog. remote gate). Bei der Komponente sind die Funktionalitäten, die für einen entfernten Zugriff erforderlich sind, (von dem entfernten Zugang bereitgestellt) getrennt von den system- und anwendungsspezifischen Funktionalitäten (von dem lokalen Zugang bereitgestellt). Die Schnittstelle der Komponente ist vorzugsweise als eine lokale Schnittstelle ausgebildet.

Mit dem erfindungsgemäßen Verfahren kann die Abarbeitungszeit eines beliebigen verteilten Computerprogramms mit mehreren Komponenten deutlich verringert werden. Das wird dadurch erzielt, dass kollozierte Zugriffe auf eine Komponente über den lokalen Zugang erfolgen und als lokale Zugriffe behandelt werden. Bei dem lokalen Zugriff können die für einen entfernten Zugriff erforderlichen zeitraubenden Funktionen somit komplett entfallen. Durch die verringerte Abarbeitungszeit ergeben sich erhebliche Kostenvorteile, da entweder mit derselben Rechenleistung wesentlich komplexere Computerprogramme oder mehr Transaktionen als bisher abgearbeitet werden können oder aber bei gleichbleibender Komplexität der Computerprogramme und Anzahl der Transaktionen die Rechenleistung der Rechner des Rechnernetzwerks reduziert werden kann.

Bei dem erfindungsgemäßen Verfahren handelt es sich um beliebig vielstufige Verfahren, d. h. von der aufgerufenen Komponente aus können nach dem vorgeschlagenen Prinzip weitere

Komponenten und von diesen wiederum weitere Komponenten u.s.w. aufgerufen werden. Gemäß einer vorteilhaften Weiterbildung der vorliegenden Erfindung wird deshalb vorgeschlagen, dass von der Komponente aus auf mindestens eine weitere Komponente über einen lokalen Zugang der weiteren Komponente zugegriffen wird, falls die weitere Komponente auf dem gleichen Rechner abgelegt ist und innerhalb der gleichen Laufzeitumgebung abläuft wie die Komponente und anderenfalls von der Komponente aus auf die mindestens eine weitere Komponente über einen entfernten Zugang der weiteren Komponente zugegriffen wird.

Um von einem kollozierten Klient (sog. collocated client) eines Rechnernetzwerks bspw. mit Client-Server-Architektur aus eine Komponente aufzurufen, greift der Klient über die Schnittstelle und den lokalen Zugang auf die Komponente zu. Innerhalb der Komponente werden dann die systemspezifischen (z. B. EJB- oder CORBA-spezifischen) Funktionalitäten und die anwendungsspezifischen Funktionalitäten (die z. B. das Ergebnis einer Rechenoperation oder den Rückgabewert eines Funktionsaufrufs berechnen) ausgeführt.

Um von einem entfernten Klient (remote client) des Rechnernetzwerks aus eine Komponente aufzurufen, greift der Klient über die Schnittstelle und den entfernten Zugang (remote gate) auf die Komponente zu. Von dem entfernten Zugang wird dann intern auf den lokalen Zugang (local gate) zugegriffen. Im Rahmen eines entfernten Zugriffs werden von

dem entfernten Zugang zunächst die für den entfernten Zugriff erforderlichen Funktionalitäten ausgeführt, bevor von dem lokalen Zugang die systemspezifischen (z. B. EJB- oder CORBA-spezifischen) Funktionalitäten ausgeführt werden. Dann erst werden schließlich die der Komponente zugeordneten anwendungsspezifischen Funktionalitäten ausgeführt.

Gemäß einer bevorzugten Ausführungsform der vorliegenden Erfindung wird über ein Proxy auf den entfernten Zugang der Komponenten zugegriffen, wobei das Proxy dieselbe Schnittstelle wie der lokale Zugang realisiert. Es wird also von einem Klient oder einer weiteren Komponente aus mittelbar über das Proxy auf den entfernten Zugang der Komponente zugegriffen. Bei einem Zugriff auf die Komponente über eine lokale Schnittstelle, die das Proxy bereitstellt, muss das Proxy zunächst die lokale Schnittstelle in eine entfernte Schnittstelle zum Zugriff auf den entfernten Zugang umsetzen. Die lokale Schnittstelle der Komponente wird also zum einen von dem lokalen Zugang realisiert (technische Implementierung) und zum anderen von dem Proxy realisiert.

Der Einsatz des Proxys führt zwar theoretisch zu einer geringfügig längeren Zugriffszeit auf entfernte Komponenten als beim Stand der Technik. Die zusätzliche Zugriffszeit ist jedoch, falls überhaupt vorhanden, im Verhältnis zu der Gesamtdauer eines entfernten Zugriffs vernachlässigbar gering und wird durch die deutlich verringerte Zugriffszeit bei

lokalen Zugriffen mehr als kompensiert. Im Mittel ergeben sich mit dem erfindungsgemäßen Verfahren für die Abarbeitung von verteilten Computerprogramme über lokale und entfernte Zugriffe auf die Komponenten deutlich geringere Abarbeitungszeiten als beim Stand der Technik.

Gemäß dieser Ausführungsform verfügt ein Klient nicht über eine direkte Referenz auf einen entfernten Zugang (remote gate) der Komponente bzw. wird er eine solche Referenz nicht benutzen, da sonst die Ortstransparenz (sog. location transparency) nicht mehr gegeben wäre, sondern er hat eine Referenz auf ein Proxy, das selbst eine Referenz auf den entfernten Zugang enthält. Das Proxy ist zwischen dem Klient und dem entfernten Zugang der aufzurufenden Komponente vorgesehen, damit der Klient stets über dieselbe Schnittstelle auf die Komponente zugreift, unabhängig davon über welchen Zugang der Zugriff erfolgt, um somit Ortstransparenz zu erhalten.

Vorteilhafterweise wird der entfernte Zugang der Komponente zur Transformation eines Parameters oder eines Ergebnisses eingesetzt, falls Dienste oder Funktionalitäten der Komponente Parameter oder Ergebnisse haben, die selbst eine Referenz auf eine weitere Komponente darstellen und die weitere Komponente bezüglich der Komponente zwar lokal, aber bezüglich des Klient entfernt angeordnet ist. Wenn bspw. eine lokale Referenz auf eine erste Komponente an eine zweite Komponente weitergegeben

werden soll, die nicht kolloziert ist, muss der entfernte Zugang die lokale Referenz in ein Proxy transformieren, das auf den entfernten Zugang der entsprechenden Komponente verweist.

Des weiteren wird vorgeschlagen, dass ein Proxy zur Transformation eines Parameters oder eines Ergebnisses eingesetzt wird, falls Dienste oder Funktionalitäten der Komponente Parameter oder Ergebnisse haben, die selbst eine mittelbare Referenz über einen weiteren Proxy auf eine weitere Komponente darstellen und diese bezüglich der Komponente zwar entfernt, aber bezüglich des Klient kolloziert angeordnet ist. Dabei sind der Klient und die weitere Komponente in dem gleichen Rechner bzw. Rechnernetzwerknoten und in der gleichen Laufzeitumgebung angeordnet.

Gemäß einer anderen vorteilhaften Weiterbildung der vorliegenden Erfindung wird vorgeschlagen, dass zum Zugriff auf eine Komponente zunächst auf einen lokalen Namensdienst (sog. naming and directory service) zugegriffen und von diesem eine Referenz auf die aufzurufende Komponente übergeben wird, wobei die Referenz auf einen lokalen Zugang der Komponente verweist, falls die aufzurufende Komponente eine kollozierte Komponente ist, und die Referenz evtl. über ein Proxy auf einen entfernten Zugang der Komponente verweist, falls die aufzurufende Komponente eine entfernte Komponente ist. Bei dem Namensdienst handelt es sich um ein lokales Verzeichnis, in

dem die Namen der Komponenten des Computerprogramms und lokale Referenzen auf die Komponenten abgelegt sind. Aus den lokalen Referenzen können die entfernten Referenzen gewonnen werden. Die entfernten Referenzen können aber auch zusätzlich zu den lokalen Referenzen in dem Namensdienst abgelegt sein.

Gemäß einer weiteren bevorzugten Ausführungsform der vorliegenden Erfindung wird vorgeschlagen, dass zum Zugriff auf eine Komponente zunächst auf einen lokalen Namensdienst zugegriffen und von diesem eine Referenz auf eine Fabrik (sog. factory) der aufzurufenden Komponente übergeben wird, wobei die Referenz auf einen lokalen Zugang der Fabrik verweist, falls die Fabrik und die aufzurufende Komponente kolloziert sind, und die Referenz ggf. über ein Proxy auf einen entfernten Zugang der Fabrik verweist, falls die Fabrik und die aufzurufende Komponente entfernt sind, und von der Fabrik eine weitere Referenz auf die aufzurufende Komponente übergeben wird, wobei die weitere Referenz auf einen lokalen Zugang der Komponente verweist, falls die Fabrik und die aufzurufende Komponente kollozierte sind, und die weitere Referenz ggf. über ein Proxy auf einen entfernten Zugang der Komponente verweist, falls die Fabrik und die aufzurufende Komponente entfernt sind. Bei EJB wird die Fabrik als Home-Interface bezeichnet. Eine Fabrik ist in der Regel erforderlich, wenn eine Komponente (z.B. Kontokomponente) mehr als eine Instanz (z.B. verschiedene Konten) aufweist. Die Referenzen aus dem lokalen Namensdienst müssen also nicht

unbedingt unmittelbar auf einen lokalen oder über ein Proxy auf einen entfernten Zugang einer aufzurufenden Komponente zeigen. Es ist vielmehr auch denkbar, dass die Referenzen zunächst auf die Fabrik der aufzurufenden Komponente zeigen. Die Fabrik verfügt ebenfalls über einen lokalen und einen entfernten Zugang. Je nach dem, ob die aufzurufende Komponente auf dem gleichen Rechner und in der gleichen Laufzeitumgebung realisiert ist wie der aufrufende Klient oder nicht, zeigt die Referenz auf den lokalen bzw. ggf. über ein Proxy auf den entfernten Zugang der Fabrik. In der Fabrik sind weitere Referenzen auf die lokalen Zugänge der entsprechenden Instanzen der aufzurufenden Komponenten abgelegt. Aus den lokalen Referenzen können die entfernten Referenzen gewonnen werden. Die entfernten Referenzen können aber auch in der Fabrik abgelegt sein. Die Fabrik übergibt entweder die lokale Referenz oder eine Referenz auf das Proxy an den aufrufenden Klient, der über die Referenz dann die aufzurufende Komponente aufruft.

Als eine weitere Lösung der Aufgabe der vorliegenden Erfindung wird ausgehend von dem Computerprogramm der eingangs genannten Art vorgeschlagen, dass die Komponente einen lokalen Zugang (local gate) zum Zugriff auf die Komponente von dem kollozierten Klient aus und einen entfernten Zugang (remote gate) zum Zugriff auf die Komponente von dem entfernten Klient aus aufweist.



Die Komponente des erfindungsgemäßen Computerprogramms steht sowohl einem kollozierten als auch einem entfernten Klient zur Verfügung. Falls der Klient in dem gleichen Rechner bzw. Rechnernetzwerkknotten und in der gleichen Laufzeitumgebung (sog. execution environment) angeordnet ist wie die aufzurufende Komponente, wird er als ein kollozierter Klient (sog. colocated client) bezeichnet. Falls der Klient in einem anderen Rechner bzw. Rechnernetzwerkknotten oder in einer anderen Laufzeitumgebung angeordnet ist als die Komponente, wird er als ein entfernter Klient (sog. remote client) bezeichnet. Der entscheidende Vorteil des erfindungsgemäßen Computerprogramms besteht darin, dass aufgrund der besonderen Ausgestaltung der Schnittstelle der Komponente mit einem lokalen Zugang und einem entfernten Zugang echte lokale Zugriffe auf die Komponente überhaupt erst möglich sind. Dabei handelt es sich im Unterschied zu den aus dem Stand der Technik bekannten kollozierten Zugriffen, die ähnlich wie entfernte Zugriffe betrachtet und abgearbeitet werden, um echte lokale Zugriffe. Dadurch erfolgen lokale Zugriffe auf eine Komponente erheblich schneller und es ergeben sich deutlich kürzere Abarbeitungszeiten für das Computerprogramm.

Der Geschwindigkeitsvorteil bei einem lokalen Zugriff ergibt sich insbesondere dadurch, dass ein kollozierter Klient keine zusätzlichen für einen entfernten Aufruf erforderlichen Funktionalitäten (sog. remote invocation overhead) ausführen muss, wenn auf die Komponente im Rahmen eines lokalen Zugriffs

zugegriffen wird. Der lokale Zugang (local gate) umfasst alle systemspezifischen (z. B. EJB- oder CORBA-spezifischen) Funktionalitäten und verarbeitet Operationsaufrufe von kollozierten Klienten (collocated clients). Der entfernte Zugang (remote gate) umfasst alle für einen entfernten Zugriff erforderlichen Funktionalitäten und verarbeitet Aufrufe von entfernten Klienten (remote clients). Des weiteren verfügt das Computerprogramm über volle Ortstransparenz (sog. location transparency).

Das Computerprogramm kann auch mehrere Komponenten mit jeweils sowohl einem lokalen als auch einem entfernten Zugang aufweisen. Ebenso kann jede Komponente über mehrere Schnittstellen mit jeweils mehreren lokalen und entfernten Zugängen verfügen. Die Komponenten können in beliebigen verteilten Systemumgebungen, wie bspw. EJB oder CORBA, realisiert werden. Mit der vorliegenden Erfindung kann bei beliebigen Systemen der zusätzliche Aufwand für die für einen entfernten Aufruf erforderliche Funktionalitäten (remote invocation overhead) bei kollozierten Zugriffen beseitigt werden.

Als noch eine weitere Lösung der Aufgabe der vorliegenden Erfindung wird ausgehend von dem Speicherelement der eingangs genannten Art vorgeschlagen, dass die Komponente einen lokalen Zugang (local gate) zum Zugriff auf die Komponente von dem kollozierten Klient aus und einen entfernten Zugang (remote

gate) zum Zugriff auf die Komponente von dem entfernten Klient aus aufweist.

Als noch eine weitere Lösung der Aufgabe der vorliegenden Erfindung wird ausgehend von dem Rechner der eingangs genannten Art vorgeschlagen, dass die Komponente einen lokalen Zugang (local gate) zum Zugriff auf die Komponente von dem kollozierten Klient aus und einen entfernten Zugang (remote gate) zum Zugriff auf die Komponente von dem entfernten Klient aus aufweist.

Schließlich wird als eine weitere Lösung der Aufgabe der vorliegenden Erfindung ausgehend von dem Rechnernetzwerk der eingangs genannten Art vorgeschlagen, dass die Komponente einen lokalen Zugang (local gate) zum Zugriff auf die Komponente von dem kollozierten Klient aus und einen entfernten Zugang (remote gate) zum Zugriff auf die Komponente von dem entfernten Klient aus aufweist.

Weitere Merkmale, Anwendungsmöglichkeiten und Vorteile der Erfindung ergeben sich aus der nachfolgenden Beschreibung von Ausführungsbeispielen der Erfindung, die in der Zeichnung dargestellt sind. Dabei bilden alle beschriebenen oder dargestellten Merkmale für sich oder in beliebiger Kombination den Gegenstand der Erfindung, unabhängig von ihrer Zusammenfassung in den Patentansprüchen oder deren Rückbeziehung sowie unabhängig von ihrer Formulierung bzw.

Darstellung in der Beschreibung bzw. in der Zeichnung. Es zeigen:

Figur 1a ein Strukturdiagramm eines Aufrufs einer Komponente eines verteilten Computerprogramms im Rahmen eines erfindungsgemäßen Verfahrens zum Abarbeiten des Computerprogramms;

Figur 1b ein Strukturdiagramm eines Aufrufs einer Komponente gemäß Figur 1a aus Sicht eines Klienten;

Figur 2 ein Strukturdiagramm eines Aufrufs einer Komponente eines verteilten Computerprogramms im Rahmen eines aus dem Stand der Technik bekannten Verfahrens zum Abarbeiten des Computerprogramms;

Figur 3a ein Strukturdiagramm eines Zugriffs auf eine kollozierte Komponente eines verteilten Computerprogramms über einen Namensdiest und eine Fabrik und einen lokalen Aufruf;

Figur 3b ein Strukturdiagramm eines Zugriffs auf eine entfernte Komponente eines verteilten Computerprogramms über einen Namensdiest und eine Fabrik und einen entfernten Aufruf;

Figur 4a ein erstes Szenario des erfindungsgemäßen Verfahrens mit einem kollozierten Klient und einer Referenz auf eine kollozierte Komponente;

Figur 4b das Szenario aus Figur 4a mit einer Referenz des Klienten auf die kollozierte Komponente;

Figur 5a ein zweites Szenario des erfindungsgemäßen Verfahrens mit einem kollozierten Klient und einer Referenz auf eine entfernte Komponente;

Figur 5b das Szenario aus Figur 5a mit einer Referenz des Klienten auf die entfernte Komponente;

Figur 6a ein drittes Szenario des erfindungsgemäßen Verfahrens mit einem entfernten Klient und einer Referenz auf eine kollozierte Komponente;

Figur 6b das Szenario aus Figur 6a mit einer Referenz des Klienten auf die kollozierte Komponente;

Figur 7a ein viertes Szenario des erfindungsgemäßen Verfahrens mit einem entfernten Klient und einer Referenz auf eine entfernte Komponente;

Figur 7b das Szenario aus Figur 7a mit einer Referenz des Klienten auf die entfernte Komponente;

Figur 8a einen Sonderfall des vierten Szenarios aus Figur 7a mit einem entfernten Klient und einer Referenz auf eine entfernte Komponente, wobei die Komponente kolloziert zu dem Klient ist; und

Figur 8b das Szenario aus Figur 8a mit einer Referenz des Klienten auf die kollozierte Komponente.

Aus dem Stand der Technik sind Computerprogramme mit mehreren Komponenten bekannt, die einzeln oder zu mehreren auf verteilt angeordneten Rechnern eines Rechnernetzwerks z. B. mit einer Klient/Server-Architektur ablauffähig sind. Die Komponenten sind bspw. als sog. Enterprise Java Beans (EJB) oder als Common Object Request Broker Architecture (CORBA) verteilte Objekte oder Komponenten ausgebildet. Eine aus dem Stand der Technik bekannte Komponente 1 ist in Figur 2 dargestellt. Die Komponente 1 ist auf einem Rechner bzw. Rechnernetzwerkknoten 2 des Rechnernetzwerks innerhalb einer bestimmten Software-Laufzeitumgebung angeordnet. Die Komponente 1 umfasst eine entfernte Schnittstelle (remote interface) 3 mit einem entfernten Zugang (remote gate) 4.

Die Komponente 1 weist verschiedene von der Systemumgebung (z.B. EJB-spezifische oder CORBA-spezifische Funktionen) bereitgestellte Funktionalitäten 5 und anwendungsspezifische Funktionalitäten 6 auf, die den der Komponente 1 zugeordneten Funktionen entsprechen. Außerdem umfasst die Komponente 1 für

einen entfernten Zugriff über den entfernten Zugang 4 erforderliche Funktionalitäten 7. Zur Ausführung der anwendungsspezifischen Funktionen der Komponente 1 im Rahmen der Abarbeitung des Computerprogramms wird auf die Komponente 1 von einem kollozierten Klient 8 oder einem entfernten Klient 9 zugegriffen. Wenn der Klient und die aufgerufene Komponente 1 auf dem gleichen Rechner 2 und innerhalb der gleichen Laufzeitumgebung realisiert sind, wird er als kollozierter (sog. colocated) Klient 8 bezeichnet. Anderenfalls wird der Klient als entfernter (sog. remote) Klient 9 bezeichnet.

Auf die Komponente 1 kann mittels eines kollozierten Zugriffs (von dem kollozierten Klient 8 aus) oder mittels eines entfernten Zugriffs (von dem entfernten Klient 9 aus) zugegriffen werden. Ein Zugriff von einem entfernten Klient 9 aus benötigt an sich wesentlich mehr Zeit als ein Zugriff von einem kollozierten Klient 8 aus, da im Rahmen des entfernten Zugriffs u.a. für einen entfernten Zugriff erforderliche Funktionalitäten 7, insbesondere Transformationen und Rücktransformationen von Parametern und Ergebnissen zum Zwecke der Datenübertragung zwischen dem Rechner 2, auf dem die Komponente 1 realisiert ist, und einem Rechner 10, auf dem der Klient 9 angeordnet ist, durchgeführt werden müssen. Nach dem Stand der Technik müssen diese zusätzlichen Funktionalitäten 7 selbst bei einem kollozierten Zugriff auf die Komponente 1 ausgeführt werden, da kollozierte Zugriffe auf die Komponente 1 über den entfernten Zugang 4 erfolgen. Somit werden also

auch kollozierte Zugriffe praktisch wie entfernte Zugriffe behandelt und erfordern entsprechend viel Rechenzeit.

Die in Figur 1a dargestellte Komponente 11 eines erfindungsgemäßen Computerprogramms verfügt dagegen über mindestens eine lokale Schnittstelle (local interface) 12 mit jeweils zwei getrennten Zugängen, einem lokalen Zugang (local gate; L) 13 und einem entfernten Zugang (remote gate; R) 14. Der lokale Zugang 13 umfasst alle systemspezifischen (z. B. EJB- oder CORBA-spezifischen) Funktionalitäten 5 und verarbeitet Operationsaufrufe von kollozierten Klienten (collocated clients) 8. Der entfernte Zugang 14 umfasst alle für einen entfernten Zugriff erforderlichen Funktionalitäten 7 und verarbeitet Aufrufe von entfernten Klienten (remote clients) 9.

Durch die zwei Zugänge 13, 14 der Komponente 11 kann das Abarbeiten eines auf Mikroprozessoren von Rechnern des verteilten Rechnernetzwerks ablauffähigen Computerprogramms mit mehreren Komponenten deutlich beschleunigt werden. Erfindungsgemäß wird auf die Komponente 11 über den lokalen Zugang 13 zugegriffen, falls die Komponente 11 auf dem gleichen Rechner 2 angeordnet ist wie der kollozierte Klient 8. Anderenfalls wird von dem entfernten Klient 9 aus mittelbar über den entfernten Zugang 14 auf die Komponente 11 zugegriffen.



Die Komponente 11 kann in beliebigen verteilten Systemumgebungen, wie bspw. EJB oder CORBA, realisiert werden. Bei beliebigen Systemen kann durch die Erfindung für kollozierte Klienten der zusätzliche Aufwand für die einen entfernten Aufruf erforderlichen Funktionalitäten (remote invocation overhead) 7 beseitigt werden. Bei der Komponente 11 sind also die Funktionalitäten 7, die für einen entfernten Zugang erforderlich sind, getrennt von den systemspezifischen Funktionalitäten 5 und den anwendungsspezifischen Funktionalitäten 6. Die lokale Schnittstelle 12 wird zum einen von dem lokalen Zugang 13 realisiert (technische Implementierung) und zum anderen von einem Proxy 15 realisiert.

Um von dem kollozierten Klient (collocated client) 8 aus die Komponente 11 aufzurufen, greift der Klient 8 über die Schnittstelle 12 und den lokalen Zugang 13 unmittelbar auf die Komponente 11 zu. Innerhalb der Komponente 11 werden dann die systemspezifischen (z. B. EJB- oder CORBA-spezifischen) Funktionalitäten 5 und die anwendungsspezifischen Funktionalitäten 6 ausgeführt. Die für einen entfernten Zugriff erforderlichen Funktionalitäten 7 werden bei einem lokalen Zugriff auf die Komponente 11 nicht ausgeführt.

Um von dem entfernten Klient (remote client) 9 aus die Komponente 11 aufzurufen, greift der Klient 9 über die Schnittstelle 12 und das Proxy 15 auf den entfernten Zugang

(remote gate) 14 zu. Dort werden die für den entfernten Zugriff erforderlichen Funktionalitäten 7 ausgeführt. Dann wird über einen internen Zugriff auf den lokalen Zugang 13 zugegriffen. Dort werden dann die systemspezifischen (z. B. EJB- oder CORBA-spezifischen) Funktionalitäten 5 und die anwendungsspezifischen Funktionalitäten 6 ausgeführt.

Zwischen dem entfernten Zugang 14 und dem entfernten Klient 9 ist das Proxy 15 vorgesehen, um die lokale Schnittstelle 12 für den Klient 9 zu realisieren. Das Proxy 15 ist ein Schnittstellenumsetzer, der im vorliegenden Fall die lokale Schnittstelle 12 in eine entfernte Schnittstelle 31 umsetzt, damit der Klient 9 über die lokale Schnittstelle 12 auf den entfernten Zugang 14 der Komponente 11 zugreifen kann. Durch das Proxy 15 bleibt die Ortstransparenz (sog. location transparency) der Komponente 11 erhalten.

Figur 1b stellt die Figur 1a aus der Sicht des Klienten 8, 9 dar. Der Klient 8, 9 sieht die Komponente 11 und die lokale Schnittstelle 12, die stets dieselbe ist, unabhängig davon, ob sie direkt vom lokalen Zugang 13 oder über ein Proxy 15 und den entfernten Zugang 14 realisiert wird.

In Figur 3a ist ein Diagramm eines Zugriffs auf eine kollozierte Komponente 11 des verteilten Computerprogramms mit zugehöriger Fabrik 19 mit dem kolloziiertem Klienten 8 dargestellt. Bei EJB wird die Fabrik auch als Home-Interface

bezeichnet. Zunächst greift der kollozierte Klient 8 auf einen Namensdienst (sog. naming and directory service) 16 zu, der lokal zu dem Klient 8, d. h. auf dem gleichen Rechner 10 und innerhalb der gleichen Laufzeitumgebung, angeordnet ist. Von dem Namensdienst 16 wird eine Kopie der Referenz 37 auf einen lokalen Zugang 38 der Fabrik 19 an den Klient 8 übermittelt. Der Klient 8 hält diese als Referenz 39 zu der Fabrik 19 und ruft so ihre Dienste auf. Die Fabrik 19 hält eine Referenz 40 auf den lokalen Zugang 13 der Komponente 11. Von der Fabrik 19 wird eine Kopie der weiteren Referenz 40 auf den lokalen Zugang 13 der aufzurufenden Komponente 11 an den Klient 8 übergeben. Dieser hält diese als weitere Referenz 41 und greift über diese auf die Komponente 11 zu.

In Figur 3b ist ein Diagramm eines Zugriffs auf eine entfernte Komponente 11 des verteilten Computerprogramms mit zugehöriger Fabrik 19 mit dem entfernten Klienten 9 dargestellt. Zunächst greift der entfernte Klient 9 auf den Namensdienst zu. Von dem Namensdienst 16 wird ein Proxy 15 mit einer Referenz 20 auf einen entfernten Zugang 18 der Fabrik 19 an den Klient 9 übermittelt. Der Klient 9 hält über das Proxy 15 eine Referenz 20 zu der Fabrik 19 und ruft so ihre Dienste auf. Die Fabrik 19 hält eine Referenz 21 auf den entfernten Zugang 14 der Komponente 11. Von der Fabrik 19 wird ein Proxy 36 mit einer weiteren Referenz 22 auf den entfernten Zugang 14 der aufzurufenden Komponente 11 an den Klient 9 übermittelt.

Dieser greift dann über das Proxy 36 und die weitere Referenz 22 auf die Komponente 11 zu.

In den Figuren 4 bis 7 sind vier verschiedene Szenarien von lokalen oder entfernten Zugriffen auf die Komponente 11 und eine weitere Komponente 23 dargestellt. Die weitere Komponente 23 verfügt ebenfalls einen lokalen Zugang 24 und einen entfernten Zugang 25. Die Komponente 11 erhält von einem Klienten 8, 9 eine Referenz auf die weitere Komponente 23 entweder als Übergabeparameter eines Dienstaufrufs oder gibt sie als Rückgabeparameter oder als Ergebnis zurück und muss ggf. eine Transformation durchführen.

In dem ersten Szenario aus Figur 4a greift der kollozierte Klient 8 über den lokalen Zugang 13 auf die Komponente 11 zu. Die weitere Komponente 23 ist in dem gleichen Rechner 2 bzw. in dem gleichen Rechnernetzwerkknotten und in der gleichen Laufzeitumgebung wie die Komponente 11 angeordnet. Die Komponente 11 verfügt über eine Referenz 26 auf den lokalen Zugang 24 der kollozierten weiteren Komponente 23. Diese Referenz 26 wird an den bzw. von dem Klient 8 übergeben, der über eine Referenz 32 und den lokalen Zugang 24 auf die weitere Komponente 23 zugreift (vgl. Figur 4b). Bei dem ersten Szenario erfolgt also keine Transformation der Referenzparameter.

In dem zweiten Szenario aus Figur 5a greift der kollozierte Klient 8 über den lokalen Zugang 13 auf die Komponente 11 zu. Die weitere Komponente 23 ist in einem anderen Rechner 27 bzw. in einem anderen Rechnernetzwerkknotten oder in einer anderen Laufzeitumgebung als die Komponente 11 angeordnet. Die Komponente 11 verfügt über eine Referenz 28 auf ein Proxy 29. Das Proxy 29 setzt die lokale Referenz 28 in eine entfernte Referenz 33 auf den entfernten Zugang 25 der entfernten weiteren Komponente 23 um. Diese Referenz 28 wird an den bzw. von dem Klient 8 übergeben, der über das Proxy 29 und eine Referenz 33 und den entfernten Zugang 25 auf die weitere Komponente 23 zugreift (vgl. Figur 5b). Bei dem zweiten Szenario erfolgt ebenfalls keine Transformation der Referenzparameter, da die weitere Komponente 23 sowohl bezüglich der Komponente 11 als auch bezüglich des Klient 8 entfernt ist. Die Verbindung von dem Klient 8 zu der Komponente 11 muss nicht zwangsläufig weiter bestehen, wenn die Verbindung über die Referenz 33 aufgebaut ist. Statt über das Proxy 29 kann die Referenz 33 auch über einen anderen Proxy hergestellt werden.

In dem dritten Szenario aus Figur 6a greift der entfernte Klient 9 über das Proxy 15 und den entfernten Zugang 14 auf die Komponente 11 zu. Die weitere Komponente 23 ist in dem gleichen Rechner 2 bzw. in dem gleichen Rechnernetzwerkknotten und in der gleichen Laufzeitumgebung wie die Komponente 11 angeordnet. Die Komponente 11 verfügt über eine Referenz 26

auf den lokalen Zugang 24 der lokalen weiteren Komponente 23. Wird die Referenz 26 an den bzw. von dem Klient 9 übergeben, muss sie in eine bzw. aus einer Referenz auf ein Proxy 30, das über die Referenz 34 und den entfernten Zugang 25 auf die weitere Komponente 23 zugreift (vgl. Figur 6b), transformiert werden. Bei dem dritten Szenario werden die Referenzparameter in dem entfernten Zugang 14 transformiert, da die weitere Komponente 23 bezüglich der Komponente 11 zwar lokal, aber bezüglich des Klient 9 entfernt ist. Die Referenzparameter werden also von dem entfernten Zugang 14 von lokal zu entfernt bzw. umgekehrt transformiert. Die Verbindung von dem Klient 9 über das Proxy 15 zu der Komponente 11 muss nicht zwangsläufig weiter bestehen, wenn die Verbindung über das Proxy 30 aufgebaut ist.

In dem vierten Szenario aus Figur 7a greift der entfernte Klient 9 über das Proxy 15 und den entfernten Zugang 14 auf die Komponente 11 zu. Die weitere Komponente 23 ist in einem anderen Rechner 27 bzw. in einem anderen Rechnernetzwerkknotten oder in einer anderen Laufzeitumgebung als die Komponente 11 angeordnet. Die Komponente 11 verfügt über eine Referenz 28 auf ein Proxy 29. Dieser setzt die Referenz 28 in eine Referenz auf den entfernten Zugang 25 der entfernten weiteren Komponente 23 um. Diese Referenz 28 wird an den bzw. von dem Klient 9 übergeben, der über ein Proxy 30, eine Referenz 35 und den entfernten Zugang 25 auf die weitere Komponente 23 zugreift (vgl. Figur 7b). Bei dem vierten Szenario erfolgt

keine Transformation der Referenzparameter, da die weitere Komponente 23 sowohl bezüglich der Komponente 11 als auch bezüglich des Klient 9 entfernt ist. Die Verbindung von dem Klient 9 über das Proxy 15 zu der Komponente 11 muss nicht zwangsläufig weiter bestehen, wenn die Verbindung über die Referenz 35 aufgebaut ist.

In Figur 8a ist ein Sonderfall des in den Figuren 7a und 7b dargestellten vierten Szenarios dargestellt, bei dem die weitere Komponente 23 zwar entfernt bezüglich der Komponente 11, aber kolloziert bezüglich des Klient 9 angeordnet ist, d. h. der Klient 9 und die weitere Komponente 23 in dem gleichen Rechner 10 bzw. Rechnernetzwerkknoten und in der gleichen Laufzeitumgebung angeordnet sind. In diesem Fall muss das Proxy 15 entfernte Referenzparameter oder Ergebnisse von Operationen in lokale transformieren bzw. umgekehrt, damit der Klient 9 über den lokalen Zugang 24 auf die weitere Komponente 23 zugreifen kann.

Wenn bspw. das Proxy 15 eine Operation an dem entfernten Zugang 14 der Komponente 11 ausgelöst hat und als Ergebnis eine Referenz auf das Proxy 29 erhält, sollte es die entfernte Referenz in eine lokale Referenz umwandeln und eine lokale Referenz übergeben. Der Klient 9, der die Referenz auf das Proxy 29 erhält, würde auch ohne eine Transformation der Referenz richtig arbeiten, für den Zugriff auf die weitere Komponente 23 jedoch wesentlich länger brauchen, da der

Zugriff auf die weitere Komponente 23 als ein entfernter Zugriff bearbeitet würde. Ohne eine Transformation der Referenz müssten zusätzliche, zeitraubende für einen entfernten Aufruf erforderliche Funktionalitäten (sog. remote invocation overhead) ausgeführt werden, obwohl der Klient 9 und die weitere Komponente 23 lokal angeordnet (collocated) sind. Um dies zu verhindern, transformiert das Proxy 15 die Referenz auf den Proxy 29 in eine lokale Referenz und übergibt diese an den Klient 9.



### Patentansprüche

1. Verfahren zum Betreiben eines verteilten Rechnernetzwerks umfassend mehrere verteilt angeordnete Rechner (2, 10, 27), wobei auf einem der Rechner (2) mindestens eine auf einem Mikroprozessor des Rechners (2) ablauffähige Komponente (11) eines Computerprogramms angeordnet ist und zum Betreiben des Rechners (2) von einem kollozierten Klient (8) oder einem entfernten Klient (9) aus auf die Komponente (11) zugegriffen wird, **dadurch gekennzeichnet**, dass auf die Komponente (11) von dem kollozierten Klient (8) aus über einen lokalen Zugang (local gate, 13) der Komponente (11) zugegriffen wird, falls der Klient (8) auf dem gleichen Rechner (2) abgelegt ist und innerhalb der gleichen Laufzeitumgebung abläuft wie die Komponente (11), und anderenfalls von dem entfernten Klient (9) aus über einen entfernten Zugang (remote gate, 14) der Komponente (11) auf die Komponente (11) zugegriffen wird.
2. Verfahren zum Betreiben eines Rechners (2) eines verteilten Rechnernetzwerks umfassend den Rechner (2) und mehrere weitere verteilt angeordnete Rechner (10, 27), wobei auf dem Rechner (2) mindestens eine auf einem Mikroprozessor des Rechners (2) ablauffähige Komponente (11) eines Computerprogramms angeordnet ist und zum Betreiben des Rechners (2) von einem kollozierten Klient

(8) oder einem entfernten Klient (9) aus auf die Komponente (11) zugegriffen wird, **dadurch gekennzeichnet**, dass auf die Komponente (11) von dem kollozierten Klient (8) aus über einen lokalen Zugang (local gate, 13) der Komponente (11) zugegriffen wird, falls die Komponente (11) auf dem gleichen Rechner (2) abgelegt ist und innerhalb der gleichen Laufzeitumgebung abläuft wie der Klient (8), und anderenfalls von dem entfernten Klient (9) aus über einen entfernten Zugang (remote gate, 14) der Komponente (11) auf die Komponente (11) zugegriffen wird.

3. Verfahren nach Anspruch 1 oder 2, **dadurch gekennzeichnet**, dass von der Komponente (11) aus auf mindestens eine weitere Komponente (23) über einen lokalen Zugang (24) der weiteren Komponente (23) zugegriffen wird, falls die weitere Komponente (23) auf dem gleichen Rechner (2) abgelegt ist und innerhalb der gleichen Laufzeitumgebung abläuft wie die Komponente (11) und anderenfalls von der Komponente (11) aus auf die mindestens eine weitere Komponente (23) über einen entfernten Zugang (25) der weiteren Komponente (23) zugegriffen wird.
4. Verfahren nach einem der Ansprüche 1 bis 3, **dadurch gekennzeichnet**, dass über ein Proxy (15, 29, 30) auf den entfernten Zugang (14, 25) der Komponenten (11, 23)

zugegriffen wird, wobei das Proxy (15, 29, 30) dieselbe Schnittstelle (12) wie der lokale Zugang (13) realisiert.

5. Verfahren nach Anspruch 3 oder 4, dadurch gekennzeichnet, dass der entfernte Zugang (14) der Komponente (11) zur Transformation eines Parameters oder eines Ergebnisses eingesetzt wird, falls Dienste oder Funktionalitäten der Komponente (11) Parameter oder Ergebnisse haben, die selbst eine Referenz (26) auf die weitere Komponente (23) darstellen und die weitere Komponente (23) bezüglich der Komponente (11) zwar lokal, aber bezüglich des Klient (9) entfernt angeordnet ist.
6. Verfahren nach Anspruch 4, dadurch gekennzeichnet, dass ein Proxy (15) zur Transformation eines Parameters oder eines Ergebnisses eingesetzt wird, falls Dienste oder Funktionalitäten der Komponente (11) Parameter oder Ergebnisse haben, die selbst eine Referenz (28) auf ein weiteres Proxy (29) darstellen und die weitere Komponente (23) bezüglich der Komponente (11) zwar entfernt, aber bezüglich des Klient (9) kolloziert angeordnet ist.
7. Verfahren nach einem der Ansprüche 1 bis 6, dadurch gekennzeichnet, dass zum Zugriff auf eine Komponente (11, 23) zunächst auf einen lokalen Namensdienst (16) zugegriffen und von diesem eine Referenz (41, 42) auf die aufzurufende Komponente (11, 23) übergeben wird, wobei

die Referenz (41) auf einen lokalen Zugang (13, 24) der Komponente (11, 23) verweist, falls die aufzurufende Komponente (11, 23) eine kollozierte Komponente ist, und die Referenz (42) ggf. über einen Proxy (36) auf einen entfernten Zugang (14, 25) der Komponente (11, 23) verweist, falls die aufzurufende Komponente (11, 23) eine entfernte Komponente ist.

8. Verfahren nach Anspruch 7, dadurch gekennzeichnet, dass zum Zugriff auf eine Komponente (11, 23) zunächst auf einen lokalen Namensdienst (16) zugegriffen und von diesem eine Referenz (17, 37) auf eine Fabrik (19) der aufzurufenden Komponente (11, 23) übergeben wird, wobei die Referenz (37) auf einen lokalen Zugang (38) der Fabrik (19) verweist, falls die Fabrik (19) und die aufzurufende Komponente (11, 23) kolloziert sind, und die Referenz (17) ggf. in ein Proxy (15) verpackt auf einen entfernten Zugang (18) der Fabrik (19) verweist, falls die Fabrik (19) und die aufzurufende Komponente (11, 23) entfernt Komponente sind, und von der Fabrik (19) eine weitere Referenz (21, 40) auf die aufzurufende Komponente (11, 23) übergeben wird, wobei die weitere Referenz (40) auf einen lokalen Zugang (13, 24) der Komponente (11, 23) verweist, falls die Fabrik (19) und die aufzurufende Komponente (11, 23) kolloziert sind, und die weitere Referenz (21) ggf. in ein Proxy (36) verpackt auf einen entfernten Zugang (14, 25) der Komponente (11, 23)

verweist, falls die Fabrik (19) die aufzurufende Komponente (11, 23) entfernt sind.

9. Auf Mikroprozessoren von Rechnern (2, 10, 27) eines verteilten Rechnernetzwerks ablauffähiges Computerprogramm umfassend mindestens eine Komponente (11) mit mindestens einem Zugang zum Zugriff auf die Komponente (11) von einem kollozierten Klient (8), der auf dem gleichen Rechner (2) abgelegt ist und innerhalb der gleichen Laufzeitumgebung abläuft wie die Komponente (11), oder von einem entfernten Klient (9) aus, der auf einem anderen Rechner (10) abgelegt ist und/oder innerhalb einer anderen Laufzeitumgebung abläuft als die Komponente (11), **dadurch gekennzeichnet**, dass die Komponente (11) einen lokalen Zugang (local gate, 13) zum Zugriff auf die Komponente (11) von dem kollozierten Klient (8) aus und einen entfernten Zugang (remote gate, 14) zum Zugriff auf die Komponente (11) von dem entfernten Klient (9) aus aufweist.
10. Speicherelement, insbesondere Read-Only-Memory, Random-Access-Memory oder Flash-Memory, für einen Rechner (2, 10, 27) eines verteilten Rechnernetzwerks, auf dem mindestens eine Komponente (11) eines auf Mikroprozessoren von Rechnern (2, 11, 27) des Rechnernetzwerks ablauffähigen Computerprogramms abgespeichert ist, wobei die Komponente (11) mindestens

einen Zugang zum Zugriff auf die Komponente (11) von einem kollozierten Klient (8) aus aufweist, der auf dem gleichen Rechner (2) abgelegt ist und innerhalb der gleichen Laufzeitumgebung abläuft wie die Komponente (11), oder von einem entfernten Klient (9) aus, der auf einem anderen Rechner (10) abgelegt ist und/oder innerhalb einer anderen Laufzeitumgebung abläuft als die Komponente (11), **dadurch gekennzeichnet**, dass die Komponente (11) einen lokalen Zugang (local gate, 13) zum Zugriff auf die Komponente (11) von dem kollozierten Klient (8) aus und einen entfernten Zugang (remote gate, 14) zum Zugriff auf die Komponente (11) von dem entfernten Klient (9) aus aufweist.

11. Rechner eines verteilten Rechnernetzwerks mit einem Speicherelement, insbesondere einem Read-Only-Memory, einem Random-Access-Memory oder einem Flash-Memory, auf dem mindestens eine Komponente (11) eines auf Mikroprozessoren von Rechnern (2, 11, 27) des Rechnernetzwerks ablauffähigen Computerprogramms abgespeichert ist, wobei die Komponente (11) mindestens einen Zugang zum Zugriff auf die Komponente (11) von einem kollozierten Klient (8) aus aufweist, der auf dem gleichen Rechner (2) abgelegt ist und innerhalb der gleichen Laufzeitumgebung abläuft wie die Komponente (11), oder von einem entfernten Klient (9) aus, der auf einem anderen Rechner (10) abgelegt ist und/oder

innerhalb einer anderen Laufzeitumgebung abläuft als die Komponente (11), **dadurch gekennzeichnet**, dass die Komponente (11) einen lokalen Zugang (local gate, 13) zum Zugriff auf die Komponente (11) von dem kollozierten Klient (8) aus und einen entfernten Zugang (remote gate, 14) zum Zugriff auf die Komponente (11) von dem entfernten Klient (9) aus aufweist.

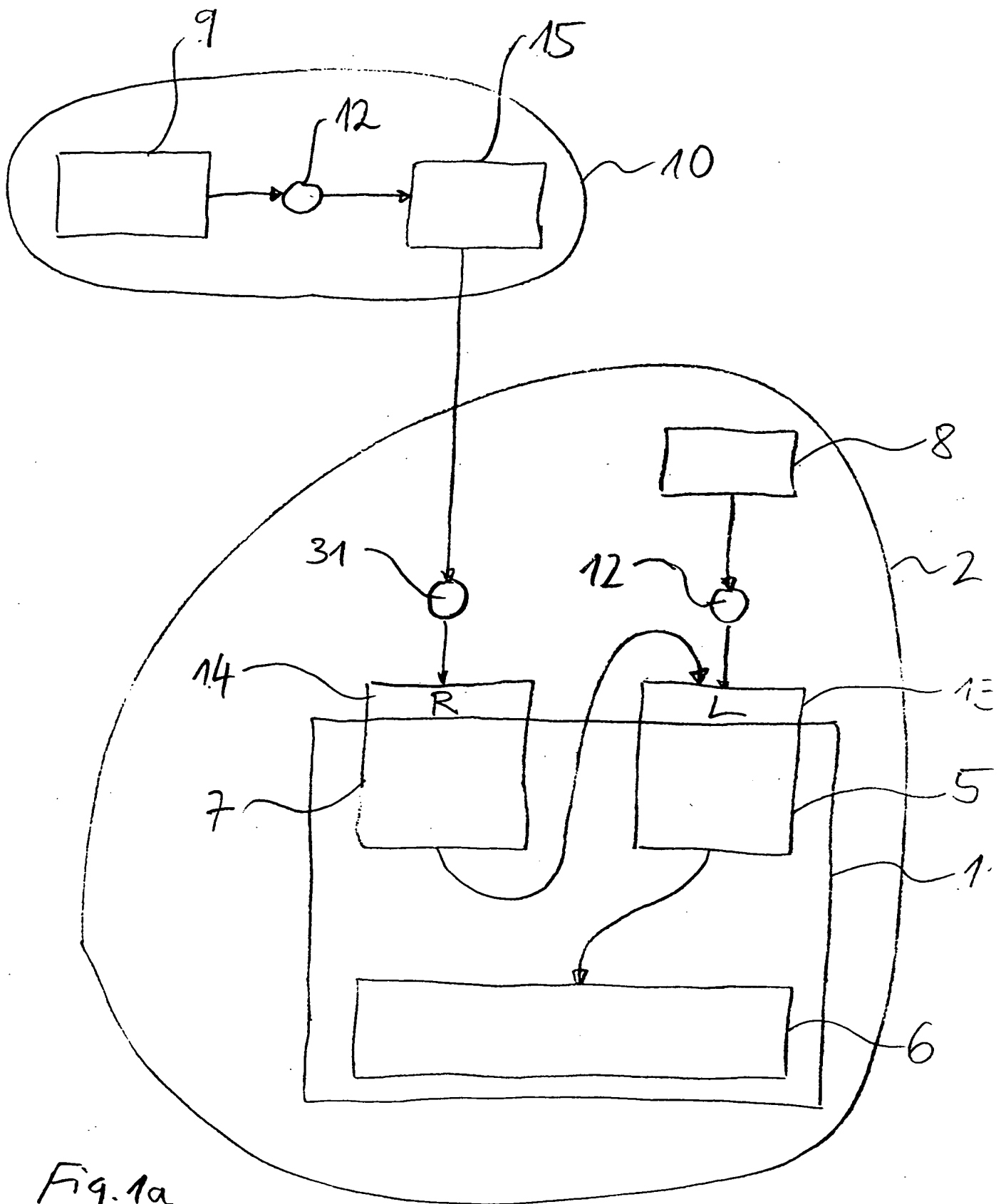
12. Verteiltes Rechnernetzwerk umfassend mehrere Rechner (2, 10, 27) mit jeweils einem Speicherelement, insbesondere einem Read-Only-Memory, einem Random-Access-Memory oder einem Flash-Memory, auf dem mindestens eine Komponente (11) eines auf Mikroprozessoren von Rechnern (2, 11, 27) des Rechnernetzwerks ablauffähigen Computerprogramms abgespeichert ist, wobei die Komponente (11) mindestens einen Zugang zum Zugriff auf die Komponente (11) von einem kollozierten Klient (8) aus aufweist, der auf dem gleichen Rechner (2) abgelegt ist und innerhalb der gleichen Laufzeitumgebung abläuft wie die Komponente (11), oder von einem entfernten Klient (9) aus, der auf einem anderen Rechner (10) abgelegt ist und/oder innerhalb einer anderen Laufzeitumgebung abläuft als die Komponente (11), **dadurch gekennzeichnet**, dass die Komponente (11) einen lokalen Zugang (local gate, 13) zum Zugriff auf die Komponente (11) von dem kollozierten Klient (8) aus und einen entfernten Zugang (remote gate,

14) zum Zugriff auf die Komponente (11) von dem entfernten Klient (9) aus aufweist.



### Zusammenfassung

Die Erfindung betrifft ein Verfahren zum Betreiben eines verteilten Rechnernetzwerks. Das Rechnernetzwerk weist mehrere verteilt angeordnete Rechner (2, 10, 27) auf, wobei auf einem der Rechner (2) eine auf einem Mikroprozessor des Rechners (2) ablauffähige Komponente (11) eines Computerprogramms angeordnet ist. Zum Betreiben des Rechners (2) wird von einem kollozierten Klient (8) oder einem entfernten Klient (9) aus auf die Komponente (11) zugegriffen. Um das Abarbeiten des Computerprogramms zu beschleunigen, wird vorgeschlagen, dass auf die Komponente (11) von dem kollozierten Klient (8) aus über einen lokalen Zugang (local gate, 13) der Komponente (11) zugegriffen wird, falls die Komponente (11) auf dem gleichen Rechner (2) abgelegt ist und innerhalb der gleichen Laufzeitumgebung abläuft wie der Klient (8), und anderenfalls von dem entfernten Klient (9) aus ggf. über ein Proxy über einen entfernten Zugang (remote gate, 14) der Komponente (11) auf die Komponente (11) zugegriffen wird. (Figur 1)

Fig. 1a

2110

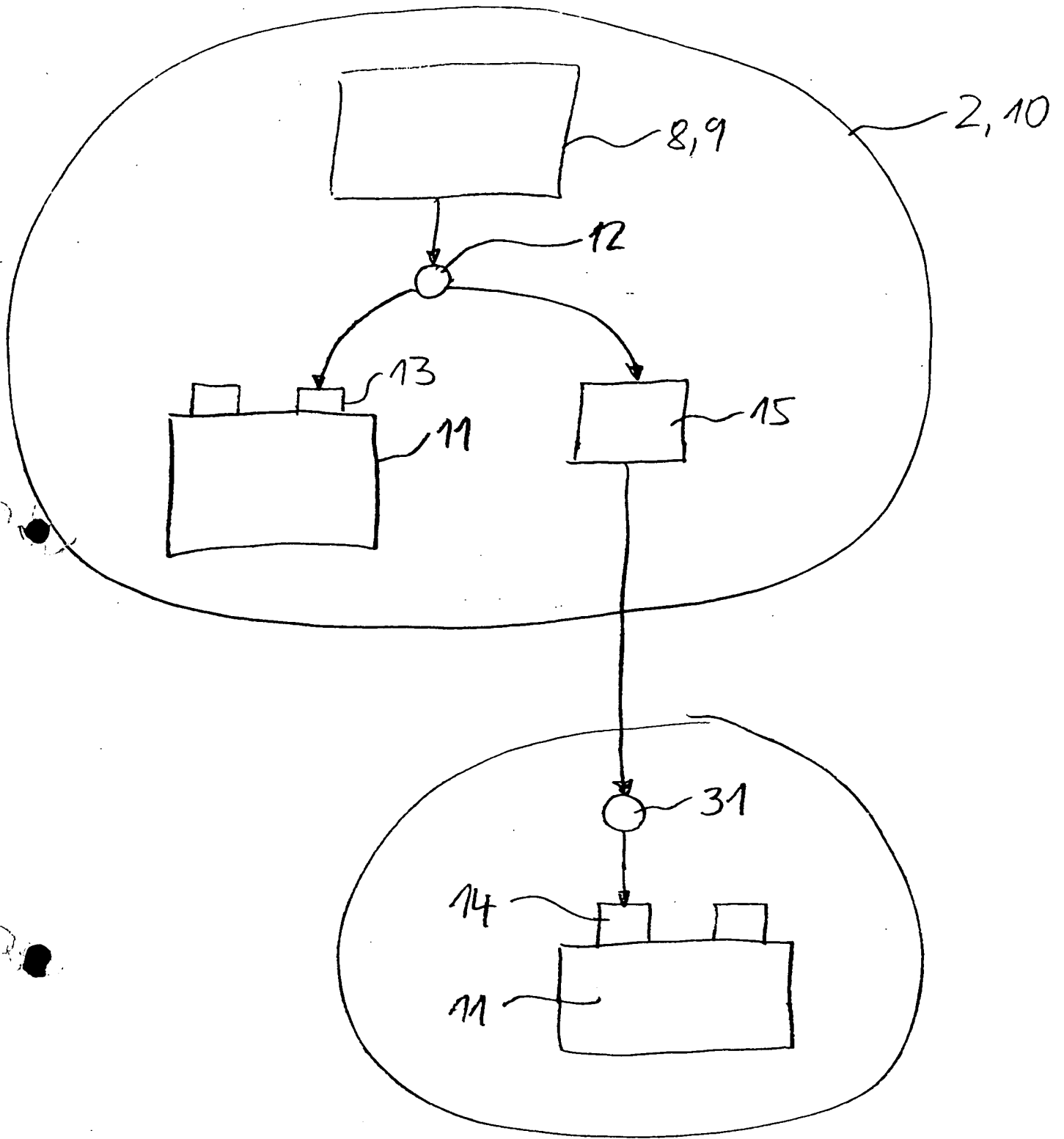
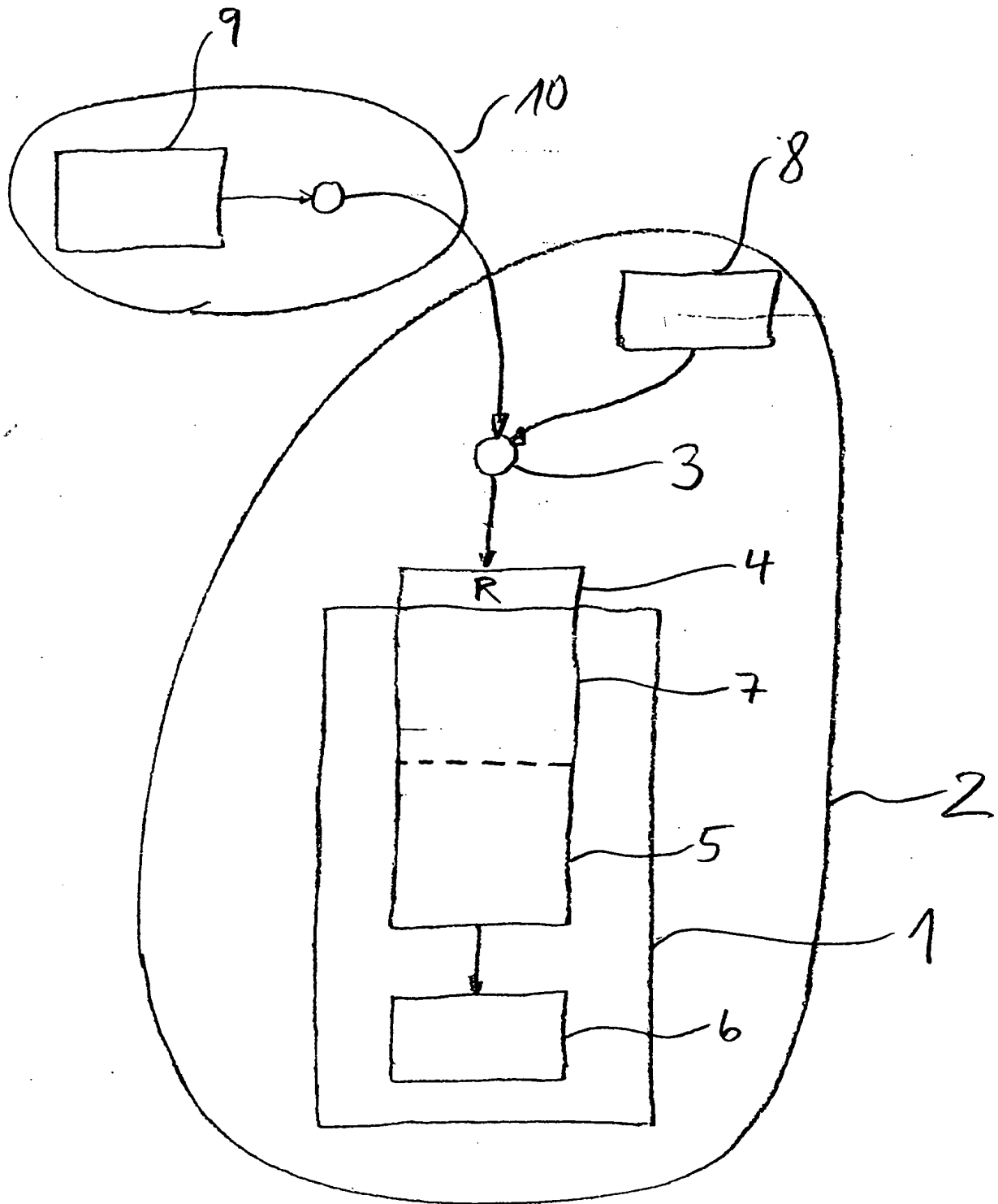


Fig. 16

3988 001

Fig. 2

4110

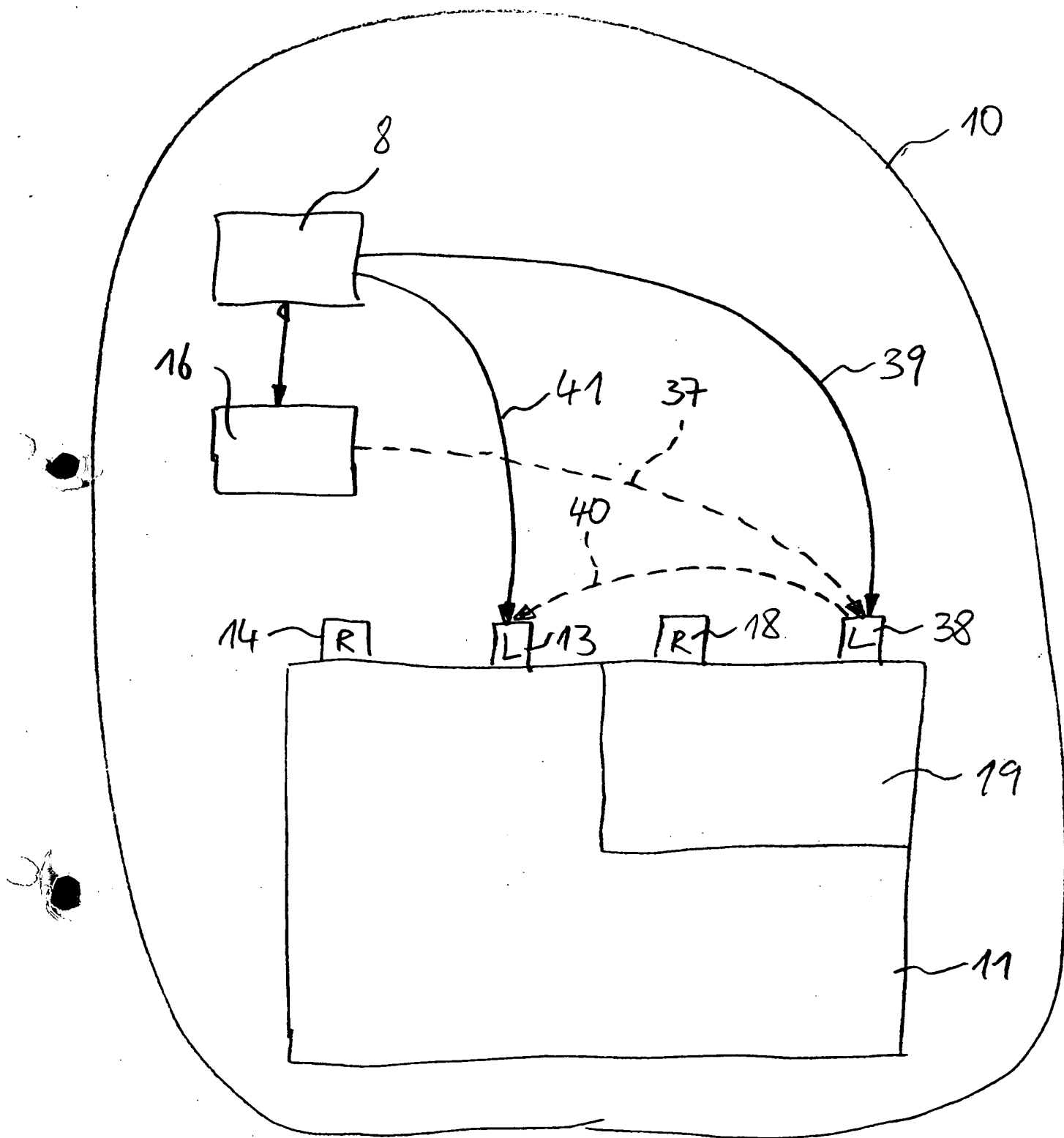
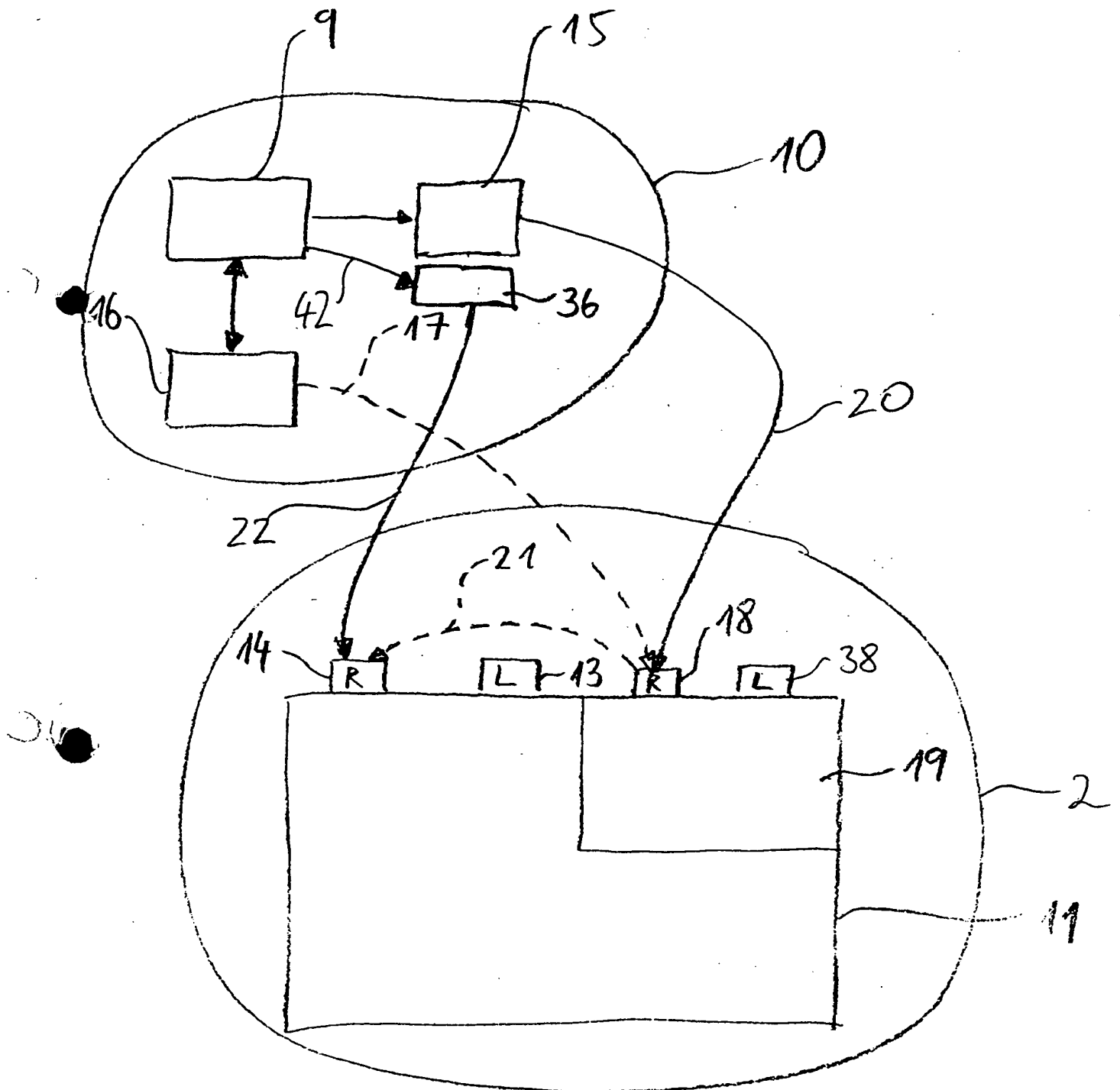


Fig. 3a

3928 001

Fig. 3b

6/10

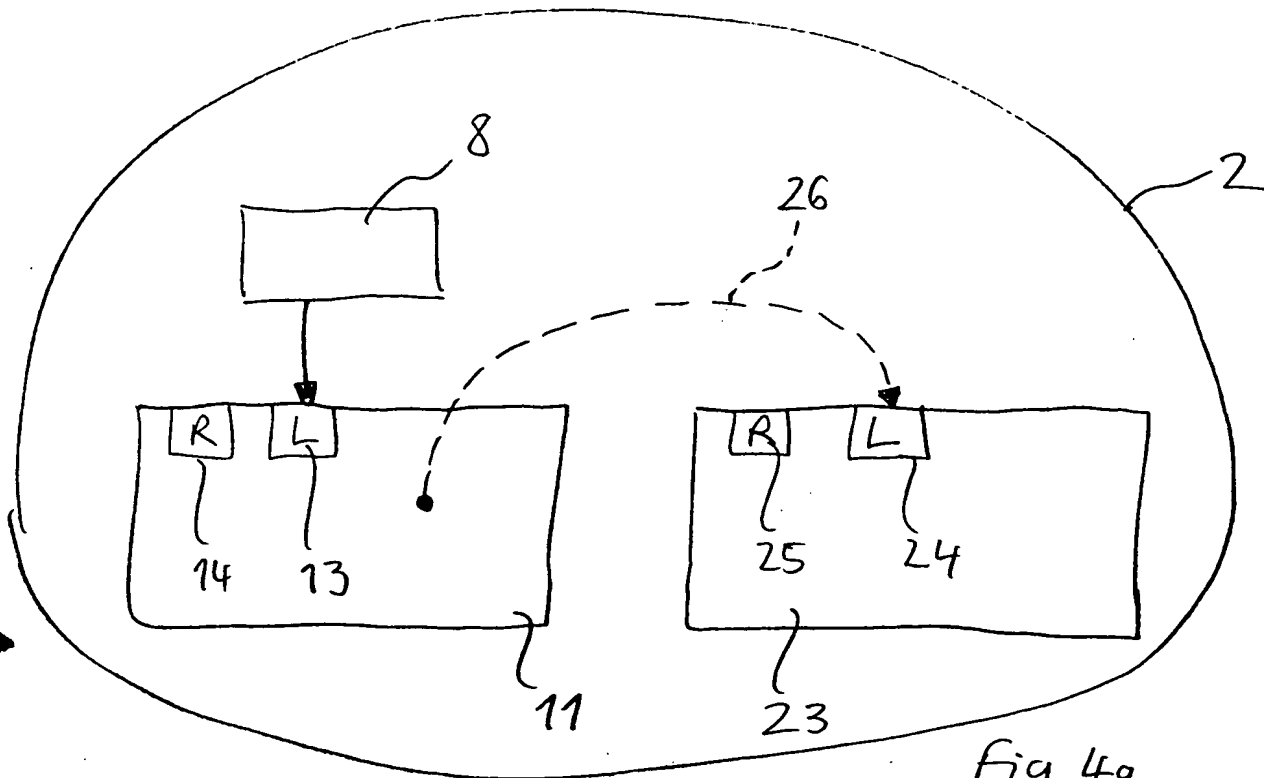


Fig. 4a

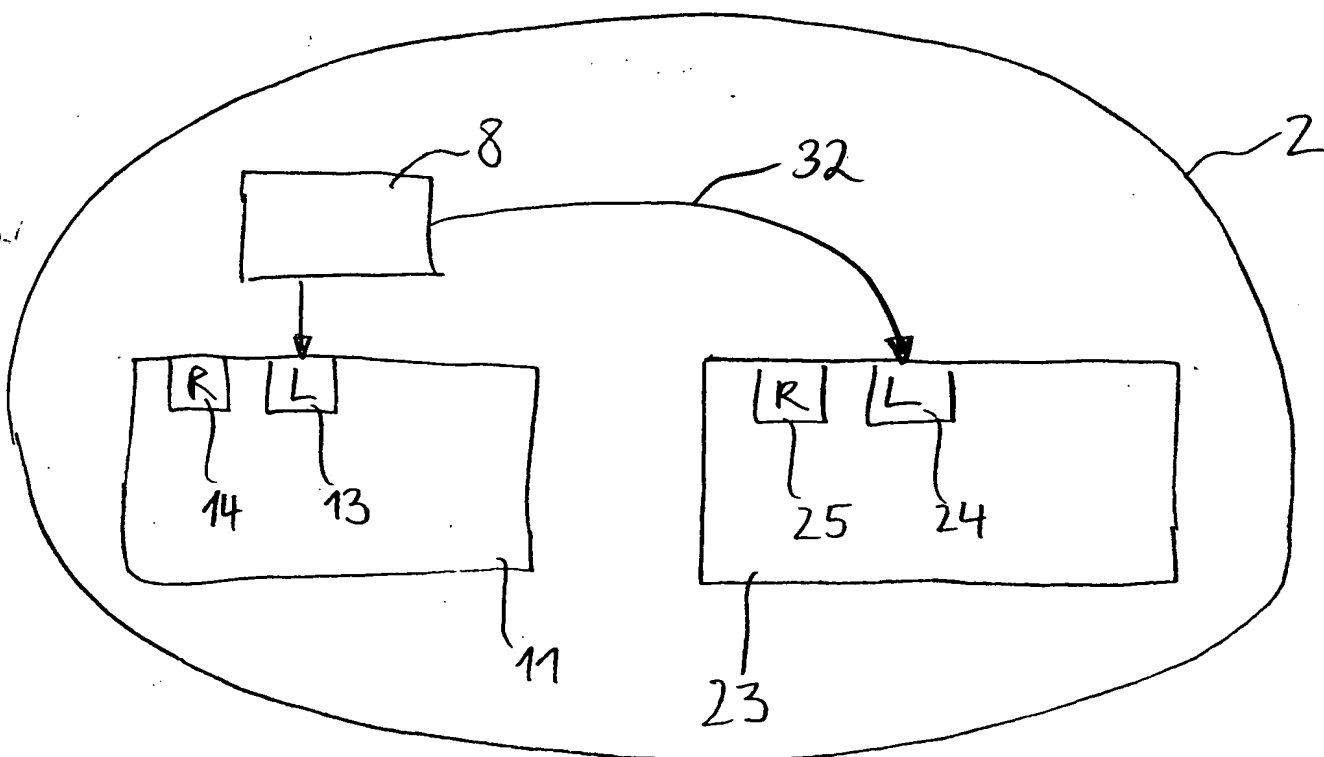


Fig. 4b

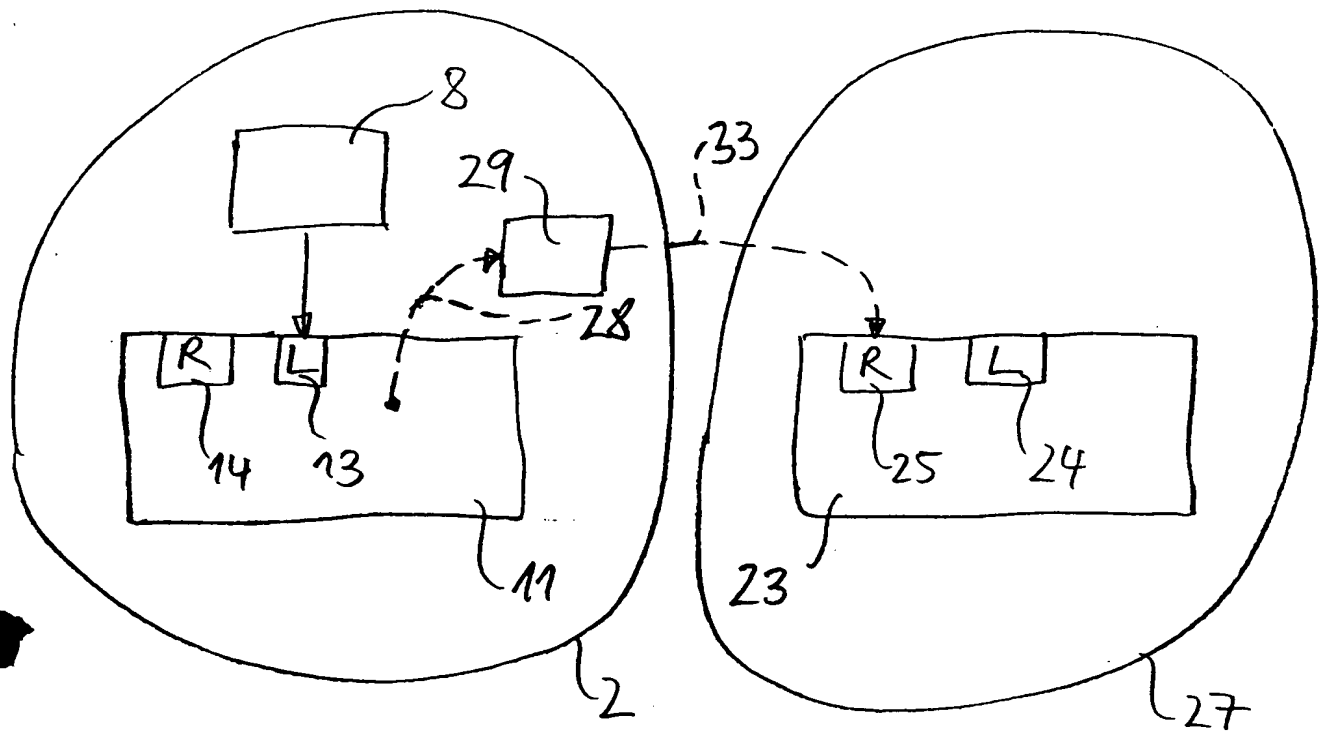


Fig. 5a

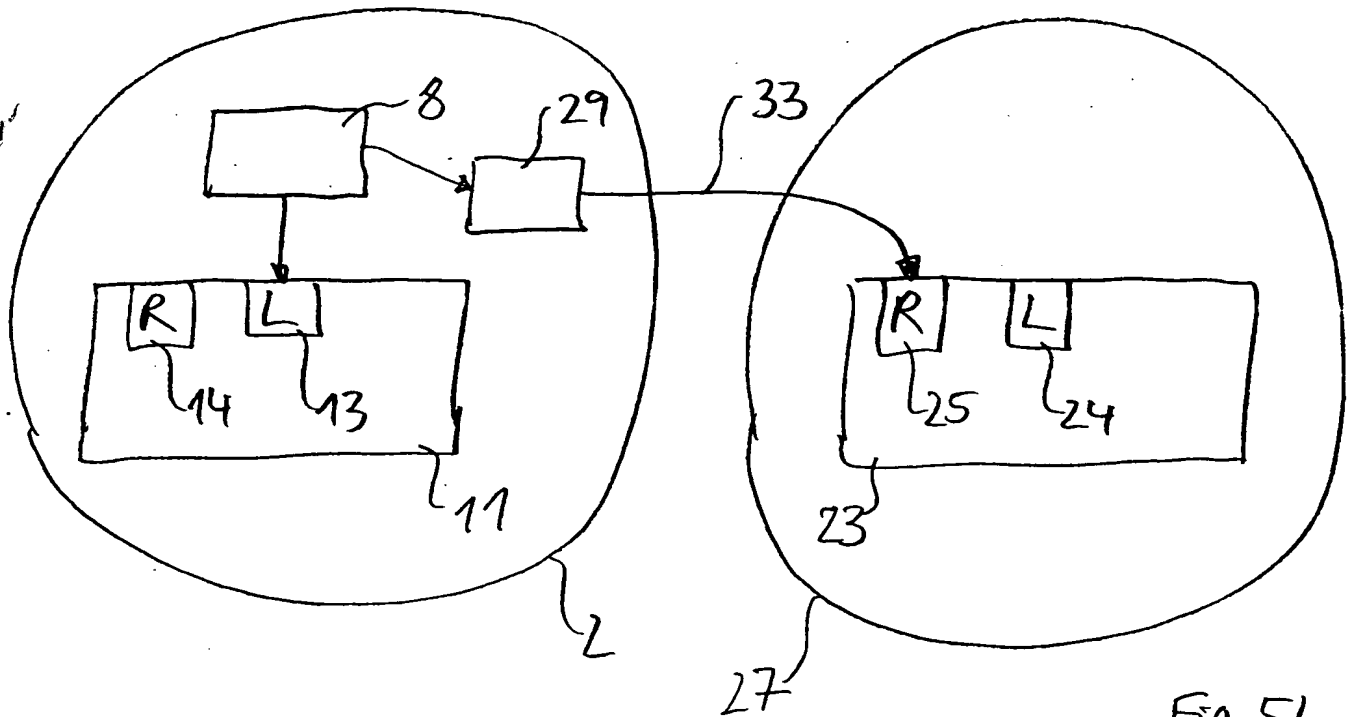
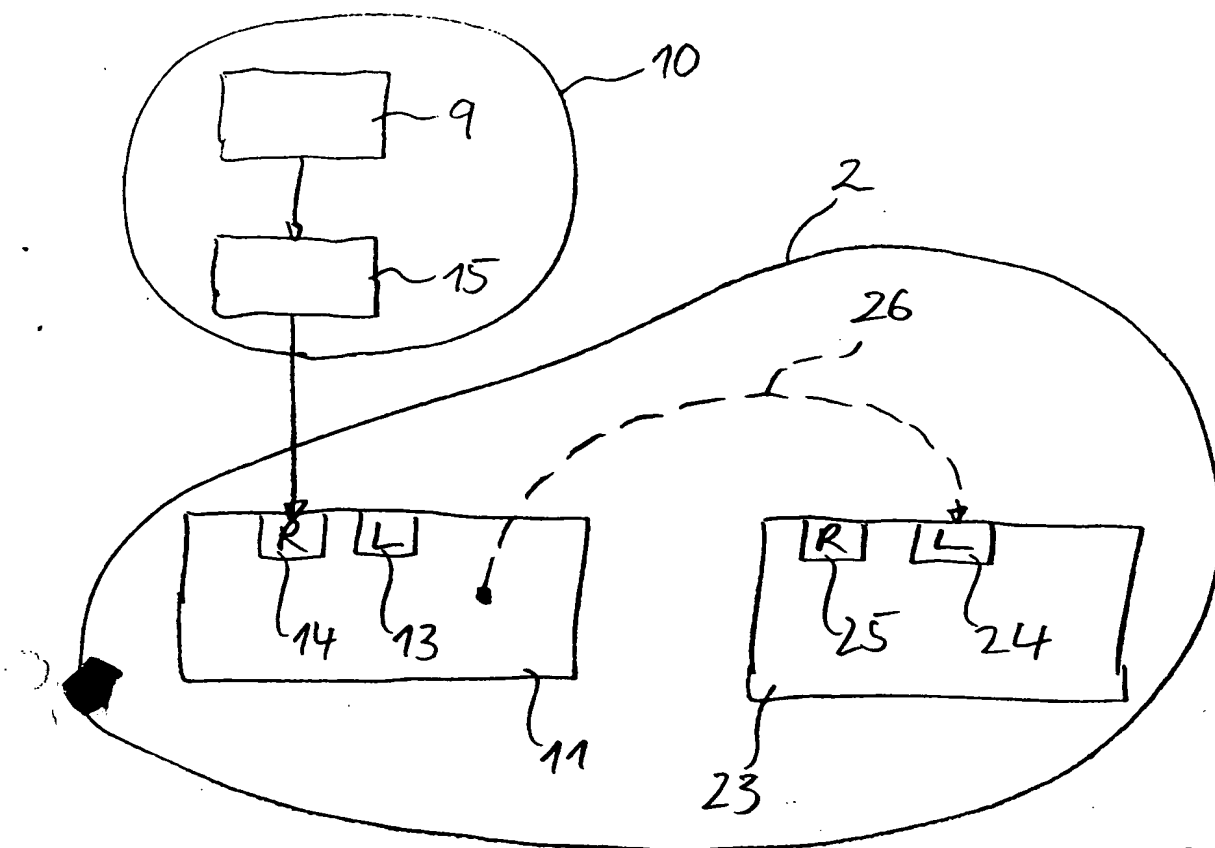
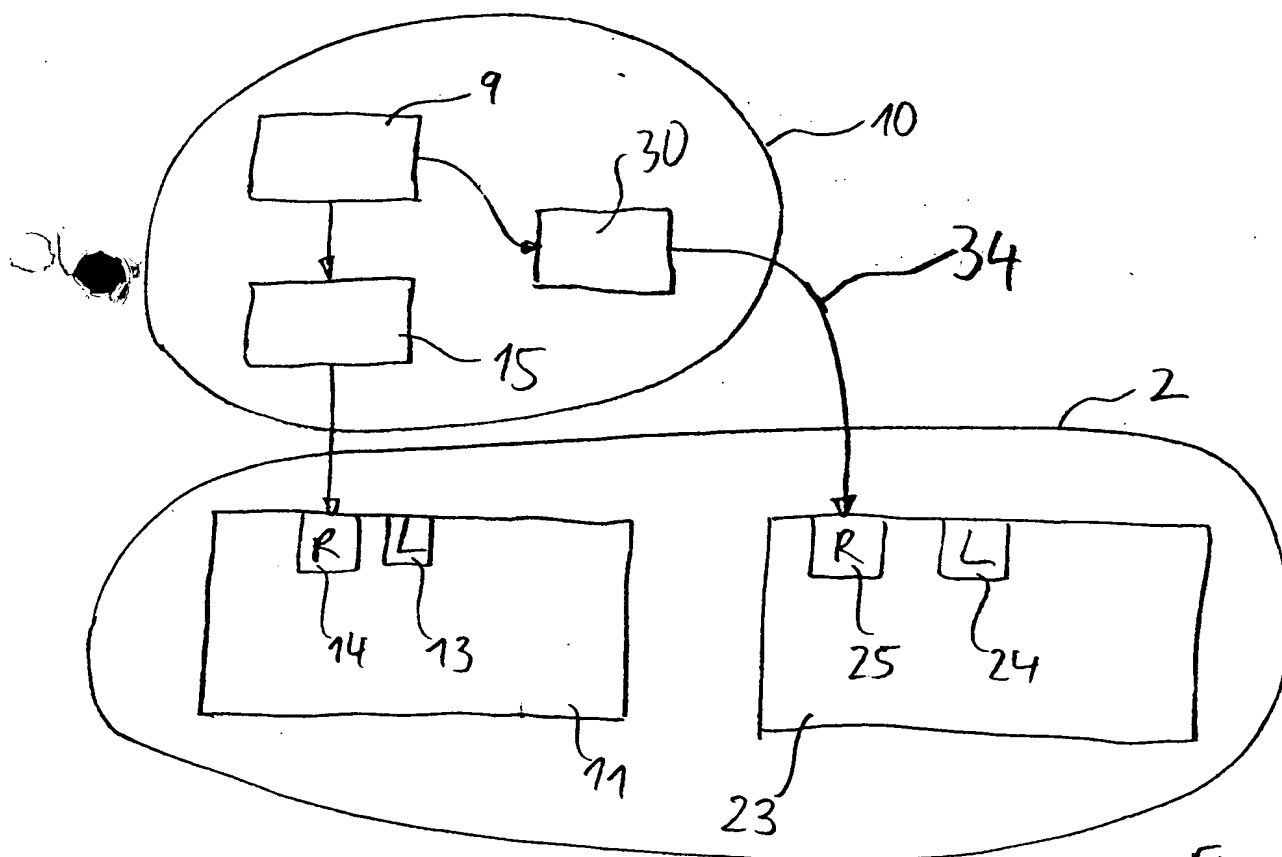


Fig. 5b



Fig. 6aFig. 6b

